# EV Assisted Resource Sharing Framework For Stream Data Processing

By

**Bareera Anam**

**00000327976**

**MS IT - 2K20**

Supervisor

**Dr Farzana Jabeen**

**Department of Computing**

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

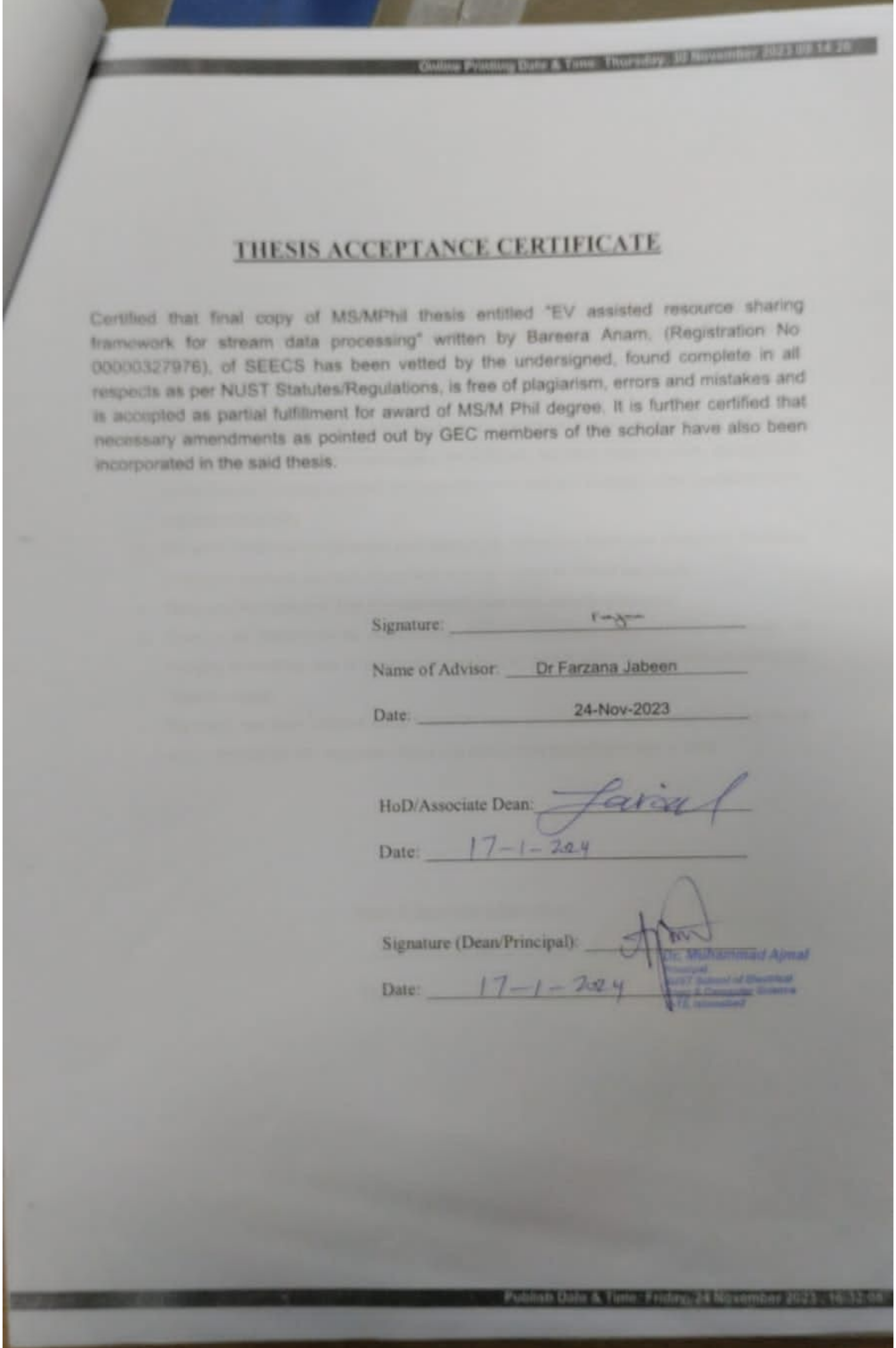
In

School of Electrical Engineering & Computer Science ,

National University of Sciences and Technology,

Islamabad, Pakistan.

(February 2022)



# THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "EV assisted resource sharing framework for stream data processing" written by Bareera Anam, (Registration No 00000327976), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Advisor: Dr Farzana Jabeen

Date: 24-Nov-2023

HoD/Associate Dean: [illegible]

Date: 17-1-2024

Signature (Dean/Principal): [illegible]

Dr. Muhammad Ajmal
Principal
NUST School of Electrical
Engg & Computer Science
H-12, Islamabad

Date: 17-1-2024

# Approval

It is certified that the contents and form of the thesis entitled "EV assisted resource sharing framework for stream data processing" submitted by Bareera Anam have been found satisfactory for the requirement of the degree

Advisor : Dr Farzana Jabeen

Signature: [illegible]

Date: 24-Nov-2023

Committee Member 1:Mr Bilal Ali

Signature: [illegible]

24-Nov-2023

Committee Member 2:Dr Mehvish Rashid

Signature: [illegible]

Date: 24-Nov-2023

Signature: ________________________

Date: ___________________________

# Dedication

This thesis is dedicated to all the deserving children who do not have access to quality education especially young girls.

# Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at Department of Computing at School of Electrical Engineering & Computer Science or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at School of Electrical Engineering & Computer Science or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Bareera Anam**

Signature: Bareera

# Contents

# Acknowledgments

Glory be to Allah (S.W.A), the Creator, the Sustainer of the Universe. Who only has the power to honor whom He pleases, and to abase whom He pleases. Verily no one can do anything without His will. From the day, I came to NUST till the day of my departure, He was the only one Who blessed me and opened ways for me and showed me the path to success. There is nothing that can pay back for His bounties throughout my research period to complete it successfully. Secondly, I want to thank my Parents whose continuous prayers help me to move forward in this journey. Last, but not least, I am grateful to my brother, Muhammad Ahsan Ullah, this whole journey is not possible without his support. Thank you, Bhai, for supporting me in every possible way.

**Bareera Anam**

# List of Figures

# List of Tables

# Abstract

The rapid growth of the Internet of Things (IoT), and the influx of real-time data streams from diverse sources have catalyzed the need for efficient and scalable stream data processing techniques. This thesis introduces an innovative paradigm that harnesses the untapped potential of electric vehicles (EVs) to enhance the scalability and efficiency of stream data processing systems. In resource-constrained environments such as edge devices and IoT deployments, conventional computing infrastructures often face challenges in meeting the demands of processing continuous and high-velocity data streams. EVs, equipped with robust computational capabilities and rechargeable batteries, offer a unique opportunity to alleviate these challenges by acting as supplementary processing nodes. The research begins by comprehensively reviewing the landscape of stream data processing, edge computing, and vehicular networks. Existing stream processing frameworks and their challenges are analyzed, paving the way for the exploration of an innovative approach to address these limitations. The proposed framework integrates EVs into the stream data processing architecture, presenting a novel approach for task offloading and collaborative stream processing. By leveraging V2V (vehicle-to-vehicle) communication, EVs within the network can dynamically share computational workloads, thereby enhancing the scalability and responsiveness of the entire system. A detailed network model is formulated to illustrate the interactions and resource sharing mechanisms among EVs, further substantiated by an in-depth system model that quantifies energy consumption, task execution times, and offloading strategies. Through meticulous simulation and performance evaluation, the novel approach is compared against conventional methods. Results reveal a substantial improvement in task completion rates, marked reduction in failure rates, and optimized energy utilization when leveraging EVs for stream data processing. The study not only underscores the efficiency of the proposed approach but also highlights its potential to revolutionize real-time data analysis in dynamic vehicular environments. This thesis contributes to the fields of stream data processing, edge computing, and vehicular networks by introducing a pioneering solution that leverages EVs to enhance the scalability, efficiency, and resilience of stream data processing systems. The findings underscore the viability of incorporating EVs as computational resources, opening new avenues for addressing the challenges posed by the burgeoning influx of real-time data streams in the era of IoT.

Chapter 1

# Introduction and Motivation

## 1.1 Introduction

With the advent of the Internet of Things (IoT) and the exponential growth of data, processing stream data efficiently and at scale has become a critical need. Data in the form of streams are produced in real time by a wide variety of devices and applications, including sensors, social media feeds, financial transactions, and network logs [1]. In many fields, such as economics, transportation, healthcare, and environmental monitoring, real-time data analysis is essential for drawing useful conclusions and acting promptly. Traditional batch processing technologies are no longer sufficient to solve the issues presented by the continuous nature of stream data. The opposite is true for stream data processing systems, which are built to process data in real-time to provide rapid analysis and decision-making. To deal with the massive amount and speedy nature of streaming data, these systems often employ distributed processing frameworks like Apache Storm, Apache Flink, or Apache Kafka Streams [2]. These stream processing systems, however, are extremely dependent on the available computational resources if they are to scale well and function well. Edge devices, and the IoT deployments are examples of resource-constrained contexts where processing power may be in short supply. The growing popularity of electric vehicles (EVs) also provides a rare chance to tap into the power of their batteries to handle stream data [3].

An original framework for the sharing resources among streams of data, with the use of electric vehicles, is proposed in this thesis. The goal is to investigate the viability of incorporating EVs computational capabilities and energy storage into stream processing systems to increase their scalability and efficiency. This paradigm seeks to reduce the strain on conventional computing infrastructures and address the difficulties of resource-constrained settings by making use of EVs' unused or underutilized capacities.

### 1.1.1 Background

The digital era's exponential data growth has given rise to new difficulties and possibilities in data processing. When dealing with a large amount of data created in real-time, traditional batch processing techniques are inadequate [4]. Since , data streams are typically continuous and move at a rapid rate, stream data processing has become an important strategy for dealing with this problem. Data streams are the collections of information that are created in real-time and come from a variety of sources, including but not limited to sensors, social media feeds, financial activities, and network logs [5]. Environmental monitoring, financial transaction fraud detection, and social media trend analysis are all examples of use cases for stream data. With the ability to analyze and make decisions in real-time, stream processing systems are built to handle the constant influx of data. To meet the difficulties of processing stream data, several distributed stream processing frameworks have been created. Popular ones include the Apache Software Foundation's Storm, Flink, and Kafka Streams projects. These frameworks permit scalability, fault tolerance, and low-latency processing by enabling parallel and distributed processing of data streams. They've found widespread use in everything from banking and telecom to logistics and healthcare.

However, the efficiency and scalability of such stream processing systems are highly dependent on the accessibility of computing power. Computing power is often in short supply in low-resource settings, such as edge devices and IoT deployments. High demands for stream processing in such situations could overwhelm conventional computing infrastructures. At the same time, the widespread uptake of EVs provides a one-of-a-kind chance to investigate other potential computational resources for stream data processing [6]. EVs, which run on rechargeable batteries, include powerful computing capabilities thanks to their onboard computers. Also, electric vehicles spend a lot of time parked and hooked up to charging stations, which means that a lot of resources are sitting there unused. Stream processing systems can be made more scalable and efficient by utilizing the computing power and energy storage capabilities of EVs, which together and can exchange data and coordinate their operations.

## 1.2 Smart IoT Environment

When smart gadgets and IoT technologies are combined, an intelligent and interconnected ecosystem is created. In this setting, gadgets can share information, conduct research, and streamline routine tasks to benefit productivity and the user experience. Exchange of data in real-time, smart decision-making, and ubiquitous connection are hallmarks of the smart IoT ecosystem [9]. Smart homes, healthcare, transportation, agriculture, and even industrial automation are just some of the many fields that benefit from the IoT's advanced infrastructure. In a "smart home," for instance, all of the appliances and electronics work together to provide the utmost convenience, safety, and comfort. In the medical field, wearable gadgets can track vitals and send that information in real-time to doctors and nurses, allowing for remote patient monitoring and more prompt treatment [10].

## 1.3 Internet of Vehicles (IoV)

The Internet of Vehicles (IoV) is a specific application area within the broader IoT domain that focuses on the integration of vehicles with IoT technologies. IoV aims to create a connected network of vehicles that can communicate with each other and with the surrounding infrastructure to enhance road safety, traffic management, and overall transportation efficiency [11].

## 1.4 Types of Smart Devices and IoT

### 1.4.1 Smartphones and Wearables

Smartphones have become ubiquitous devices that serve as the central hub for accessing and controlling various smart devices. They enable users to interact with IoT systems, monitor and control smart home devices, and access real-time information. Wearable devices, such as smartwatches and fitness trackers, collect data on users' health and activities, facilitating personalized monitoring and analysis.

are now underutilized [7]. Our research is predicated on the concept of incorporating EV resources into stream data processing systems. To facilitate the smooth incorporation of EVs as extra processing nodes in preexisting stream processing systems, a framework for EV-assisted resource sharing is developed. The framework's goal is to balance workloads and improve system performance by delegating duties for stream processing to both conventional compute nodes and EVs. This not only makes the stream processing system more scalable as a whole, but it also helps alleviate some of the difficulties that arise in contexts with limited resources. The system overview is presented in Figure 1.1, offering a visual depiction of the entire system's structure and components.

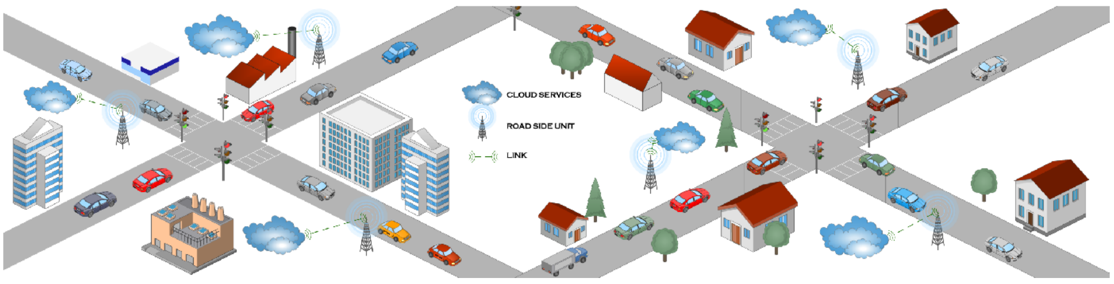


**Figure 1.1:** Visual Depiction of the system

There are several possible advantages of using EV resources to process stream data. First, it enables the effective recruitment of otherwise unused computing assets, hence increasing the system's total processing power. Second, the framework can potentially lessen the energy consumption of conventional computer infrastructures by making use of the energy storage capacities of EVs [8]. In light of mounting environmental concerns, this is in line with the rising need for eco-friendly computer options. Intelligent resource allocation algorithms must be developed before the full potential of EV-assisted resource sharing can be realized. To accomplish optimal task allocation, these algorithms should think about things like the computing capabilities of EVs, the energy levels of EV batteries, and the workload distribution. Extensive simulations and real-world experiments will be used to evaluate the efficacy of these algorithms and the performance and scalability of the proposed framework. Our relationship with technology and the physical environment has been profoundly altered by the advent of smart gadgets and the IoT. Our ability to easily collect and share information thanks to smart gadgets like smartphones, wearables, and connected sensors has made them ubiquitous in our daily lives. In contrast, the IoT describes a system in which diverse objects and systems are linked

### 1.4.2 Connected Home Devices

Connected home devices, also known as smart home devices, include smart thermostats, lighting systems, security cameras, door locks, and appliances. These devices can be controlled remotely through smartphone apps or voice assistants, allowing users to manage their homes efficiently, save energy, and improve security.

### 1.4.3 Industrial IoT (IIoT) Devices

Industrial IoT devices are employed in industrial settings for process automation, asset monitoring, and predictive maintenance. They include sensors, actuators, and controllers that collect and transmit data to optimize production processes, monitor equipment health, and enable proactive maintenance.

### 1.4.4 Connected Healthcare Devices

Connected healthcare devices encompass a wide range of IoT-enabled medical devices, including wearable health trackers, remote patient monitoring systems, smart medical implants, and telemedicine devices. These devices enhance healthcare delivery by enabling continuous monitoring, early detection of health issues, and remote consultations.

### 1.4.5 Smart City Infrastructure

Smart city infrastructure involves the deployment of IoT technologies in urban environments to enhance services and improve quality of life. It includes smart streetlights, waste management systems, parking management, environmental monitoring, and traffic management systems. These interconnected systems enable efficient resource utilization, reduce congestion, and promote sustainability.

### 1.4.6 Internet of Vehicles (IoV) Devices

IoV devices include connected vehicles, roadside sensors, and infrastructure components that enable vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. These devices facilitate real-time traffic monitoring, collision avoidance, autonomous driving, and intelligent transportation systems.

## 1.5 Motivation

The need for effective and scalable stream data processing in the age of big data and the Internet of Things (IoT) inspired this study's investigation. Sensors, social media feeds, financial transactions, and network logs are just a few of the sources that produce stream data at an unparalleled rate. Timely choices, the identification of abnormalities, and the acquisition of useful insights across several areas all rely on the results of real-time data analysis. The rising popularity of EVs provides a one-of-a-kind chance to investigate other potential computational resources for stream data processing. Electric vehicles (EVs) have impressive computational capabilities due to their high-powered onboard processors and rechargeable batteries. In addition, electric vehicles spend a lot of time parked and hooked up to charging stations, creating opportunities to make use of otherwise unused resources.

There are several convincing arguments in favor of incorporating EV resources into stream processing systems. First, it allows for the effective utilization of unused or underutilized computational resources, which increases the system's total processing capacity. This has the potential to increase the system's scalability and responsiveness to the stream of data that never stops coming in. Second, the suggested framework has the potential to lower the energy consumption of conventional computer infrastructures by capitalizing on the energy storage capabilities of EVs. As energy efficiency and environmental concerns rise in importance in today's data processing, this fits well with the growing need for sustainable computing solutions. Further, the difficulties encountered in resource-limited settings can be mitigated by including EVs in stream processing systems. The architecture can reduce the workload on conventional computing nodes and improve system performance by using electric vehicles as distributed computing and energy storage. The study's overarching goal is to design and implement a resource-sharing framework for electric vehicles that can be used with any data-streaming system. Taking into account parameters like computational capabilities, energy levels of EV batteries, and workload distribution, this framework will be created to effectively allocate stream processing duties across conventional computing nodes and EVs. Extensive simulations and actual implementations will test the proposed framework's performance and scalability. This thesis seeks to advance the state of the art in stream data processing by investigating the unrealized potential of EV-assisted resource-sharing frameworks. New perspectives on improving scalability, efficiency, and sustainability in the context of IoT and big data can be gained from the results of this study on the viability and usefulness of harnessing EV resources for stream data processing.

## 1.6 Purpose of Research

The primary goal of this research is to develop an EV-assisted resource-sharing framework for stream data processing. The framework aims to seamlessly integrate electric vehicles (EVs) as additional processing nodes into existing stream processing systems, thereby enhancing their scalability and efficiency. The purpose of this research is listed below;

- To design intelligent resource allocation algorithms that efficiently distribute stream processing tasks among traditional computing nodes and EVs. These algorithms will consider factors such as the computational capabilities of EVs, the energy levels of EV batteries, and the workload distribution to achieve optimal task allocation and load balancing.
- To explore the untapped potential of utilizing the computational capabilities and energy storage of EVs for stream data processing. By leveraging the idle or underutilized resources of EVs, the proposed framework aims to address the challenges posed by resource-constrained environments and improve the overall performance of stream processing systems.
- To explore the untapped potential of utilizing the computational capabilities and energy storage of EVs for stream data processing. By leveraging the idle or underutilized resources of EVs, the proposed framework aims to address the challenges posed by resource-constrained environments and improve the overall performance of stream processing systems.
- To evaluate the performance and scalability of the proposed EV-assisted resource-sharing framework through extensive simulations and real-world experiments. The findings will provide valuable insights into the feasibility and effectiveness of leveraging EV resources for stream data processing and offer practical guidance for implementing such frameworks in diverse application domains.

- To contribute to the advancement of stream data processing techniques by exploring innovative approaches to handle the continuous and high-velocity nature of stream data. By harnessing the computational power and energy storage of EVs, the research aims to pave the way for more sustainable, efficient, and scalable stream processing systems, enabling real-time analysis of large-scale streaming data in resource-constrained environments.

## 1.7 Problem Statement

As autonomous vehicles continue to advance in terms of features and functionalities, the issue of inadequate resources poses a challenge to their optimal performance. To address this challenge, offloading has emerged as a technique utilized by automated vehicles to optimize their computation, performance, and storage capacity. Task offloading allows automated vehicles to delegate certain tasks to neighboring vehicles for processing. While task offloading proves effective in conserving resources, static offloading strategies can be costly. Therefore, there is a need to explore dynamic and efficient offloading techniques that can enable automated vehicles to leverage the resources of neighboring vehicles for enhanced performance and resource utilization.

Chapter 2

# Literature Review

Stream processing has emerged as a critical technology for real-time data analysis and decision-making. It enables the processing of continuous data streams, providing valuable insights and timely responses. Stream processing involves the analysis and processing of data streams in real time. It differs from traditional batch processing by handling infinite, continuously arriving data. Many applications, such as financial analytics, social media monitoring, and IoT data processing, rely on stream processing for immediate insights [12]. IoT devices generate continuous streams of data from various sources such as sensors, actuators, and connected devices. These data streams often exhibit high velocity, variety, and volume, requiring immediate processing to extract meaningful information. Stream processing in IoT devices involves the real-time analysis and processing of these data streams to detect events, monitor system health, and trigger appropriate responses. However, stream processing in IoT devices faces several challenges [13]. Firstly, the sheer volume and velocity of data generated by IoT devices require efficient processing mechanisms to handle the incoming streams in real time. Additionally, data quality and reliability are crucial factors, as IoT data streams may contain noise, missing values, or inconsistencies that need to be addressed during processing. Moreover, the resource-constrained nature of many IoT devices poses challenges in terms of computational capabilities and energy efficiency. Stream processing algorithms and techniques need to be optimized to operate within the limited resources of IoT devices, ensuring minimal energy consumption and efficient utilization of processing power. To address these challenges, researchers have proposed various solutions. Edge computing, which involves performing stream processing tasks closer to the data source, can alleviate the burden on centralized cloud infrastructure and reduce latency.

Edge devices can perform initial data preprocessing, filtering, and lightweight analytics, offloading the more significant processing tasks to the cloud [14]. As IoT continues to expand and evolve, further research and innovation in stream processing techniques will be essential to unlock the full potential of IoT data and drive the development of smart and responsive IoT systems.

## 2.1 Edge Computing

In numerous cloud-based applications, data centers are commonly used as central servers for processing the data generated by edge devices like smartphones, tablets, and wearables. These edge devices serve as data sources, collecting and transmitting data to the centralized cloud infrastructure for storage, analysis, and processing. By leveraging the computational resources and scalability of data centers, cloud-based applications can handle large volumes of data and perform complex computations, providing services and insights to edge devices and their users [15]. The research on edge computing is advancing swiftly as a result of the constraints imposed by traditional cloud platforms. Recognizing the limitations of relying solely on centralized cloud infrastructure, researchers are actively exploring edge computing as a promising alternative. By moving computational resources closer to the edge of the network, edge computing offers advantages such as reduced latency, improved responsiveness, and enhanced privacy and security. This shift towards edge computing is driven by the need to overcome the limitations of the cloud platform and unlock new possibilities for real-time data processing, analytics, and decision-making [16]. Edge computing plays a crucial role in the migration of data and services from centralized nodes to the network's edge. This shift is driven by the increasing demand for faster processing, reduced latency, and enhanced user experience. By bringing computational resources closer to the edge of the network, edge computing enables efficient data processing and service delivery in proximity to the data sources. This approach eliminates the need to transmit massive amounts of data to distant centralized nodes for processing, resulting in improved response times and optimized bandwidth utilization [17]. Rather than relying solely on centralized cloud servers for data analysis, this approach brings the intelligence and processing power to the edge of the network. By utilizing intelligent edge devices like Raspberry Pi, data analytics can be performed locally, closer to the data sources in the IoT ecosystem. This enables real-time or near-real-time analysis, reduces the need for extensive data transfer, and improves overall efficiency. The use of intelligent edge devices for IoT data analytics opens up new possibilities for distributed and decentralized analytics, enabling faster insights and more responsive decision-making [18].

Mobile-edge cloud computing is an emerging concept that aims to bring the power of cloud computing closer to mobile users by deploying cloud resources at the edge of radio access networks. This paradigm recognizes the importance of minimizing latency and improving the overall user experience by providing computational capabilities in close proximity to mobile devices. By leveraging the resources available at the network edge, mobile-edge cloud computing enables mobile users to offload computation tasks, access cloud services, and benefit from enhanced performance and responsiveness [19]. The Internet of Things (IoT) has transitioned from experimental technology to a critical component of future customer value in various industries. IoT, combined with intelligent and big data analytics, is poised to revolutionize sectors like healthcare, smart cities, and industrial automation. As we move towards the fifth generation of wireless communication systems, IoT's significance becomes even more pronounced. Multi-access edge computing (MEC) technology has emerged to extend cloud computing capabilities to the edge of the radio access network. This advancement enables real-time, high-bandwidth, and low-latency access to radio network resources, transforming the way data is processed and enhancing overall connectivity and performance [20].

## 2.2 Vehicular task offloading

In recent times, the rising demand for multimedia services on mobile devices, such as smartphones, iPads, and electronic tablets, has led to a significant surge in mobile data traffic [21]. This exponential growth has posed a severe burden on mobile network operators. To tackle this challenge effectively, a viable solution is to employ complementary technologies, such as small cell networks and WiFi networks, for mobile data offloading. Mobile data offloading involves diverting data traffic away from the traditional mobile network infrastructure to alternative network technologies. By utilizing complementary technologies, the goal is to alleviate the congestion and capacity limitations of the mobile network. Offloading data to small cell networks and WiFi networks allows for the efficient distribution of data traffic, reducing the strain on the primary mobile network [22]. Mobile-edge computing (MEC) in conjunction with the Internet of Vehicles (IoV) is a novel network technology that holds great potential for enhancing task computing and offloading efficiency. By bringing computational capabilities closer to the network edge, MEC enables faster and more localized processing of tasks. When integrated with IoV, which connects vehicles to the network, it allows for efficient offloading of computationally intensive tasks [23]. This combination of MEC and IoV empowers vehicles to leverage nearby edge resources, optimizing task execution and improving overall computing efficiency. Task offloading from vehicle to vehicle involves the transfer of computationally intensive tasks and large data payloads from one vehicle to another. This approach leverages the computing and storage capabilities of nearby vehicles to offload the processing burden and reduce latency. By utilizing the resources of neighboring vehicles, task offloading enables efficient execution of tasks that require substantial computational power and handling of large datasets. Vehicular edge computing (VEC) is an emerging field that aims to optimize vehicular services by distributing computational tasks between VEC servers and local vehicular terminals. With the advancement of vehicle intelligence and capabilities, there is great potential for the development of new and exciting applications. By leveraging the resources of neighboring vehicles, VEC ensures efficient utilization of network resources while alleviating the computational load on the VEC servers. However, the dynamic nature of vehicles' mobility, the changing network topology, and the unpredictable availability of computing resources pose challenges in managing VEC environments. These factors make it difficult to accurately predict and adapt to the evolving conditions, requiring innovative solutions to optimize task distribution and resource allocation in VEC systems [24].

The rapid advancements in the Internet of Vehicles (IoV) have led to the proliferation of smart vehicles equipped with computation units, multi-communication technologies, sensor platforms, and human-machine interaction devices. These technological advancements have made it increasingly feasible for these smart vehicles to offer a range of intelligent traffic applications. Examples of such applications include smart parking decisions, real-time road traffic monitoring, and automatic management systems. Furthermore, there is a wide array of multimedia onboard applications available for both passengers and drivers, enhancing the overall experience and convenience of smart vehicles. The integration of these advanced technologies in vehicles opens up a multitude of possibilities for creating innovative and efficient solutions in the realm of transportation and mobility [25]. These advanced applications require robust computational capabilities to meet their demanding requirements. Particularly, applications involving real-time interaction and video processing, such as image-assisted navigation, immersive applications, and natural language processing, have strict limitations on latency and delay. These applications necessitate high-grade computation to perform tasks in real-time and process data efficiently. Meeting these requirements is crucial to ensure seamless user experiences and enable the effective functioning of these applications. Therefore, optimizing computational resources and implementing efficient algorithms are vital to address the latency constraints and deliver smooth performance in these computationally intensive applications [26].

## 2.3 Stream Processing

In the realm of big data, the significance of streaming data has grown immensely. With the increasing demand for managing vast and continuously flowing data sets, streaming systems have reached a stage of development where they are now widely accepted and utilized by businesses. These systems have matured to a point where they offer effective solutions for processing and harnessing the potential of unbounded data streams in various domains. This paper presents the dataflow model, which is widely used in stream data processing systems. It discusses the principles and advantages of the dataflow model in enabling efficient and scalable processing of unbounded, out-of-order data streams [27]. It also covers important aspects such as event time handling, windowing, and fault tolerance. This paper introduces Storm, a popular open-source stream data processing system. It describes the architecture and features of Storm, which enables scalable and fault-tolerant real-time computation on large-scale data streams. It covers key concepts such as spouts, bolts, and topologies, and provides insights into how Storm handles reliability and parallelism [28]. In recent years, data stream processing has experienced a surge in interest and activity due to the growing Big Data landscape and the emergence of the Internet of Things (IoT). This has posed unique challenges for conventional stream processing engines such as System S, Borealis, and STREAM[4]. The need to efficiently process and analyze massive streams of data generated by IoT devices has pushed the boundaries of traditional stream processing approaches [29].

Chapter 3

# Design and Methodology

## 3.1 Network Model

The network configuration showcases a scenario where vehicles play a significant role in task offloading through stream processing. This arrangement involves leveraging the capabilities of vehicles within the network to efficiently offload computational tasks and perform stream processing operations. The vehicles act as integral components in the network architecture, contributing to the overall processing capacity and enabling seamless data flow for stream processing applications.

### 3.1.1 Electric Vehicles

Within our network model, we employed EVs that rely on onboard battery packs for propulsion. These EVs utilize rechargeable batteries both to store and provide energy for the task offloading process. These vehicles are capable of generating complex computational tasks that demand substantial processing resources. However, recognizing the constraints of available resources, they strategically delegate these resource-intensive tasks to neighboring vehicles for computation, facilitated through stream processing. The computational design for vehicles is illustrated in Figure 3.1, providing a visual representation of our vehicular network's processing architecture.

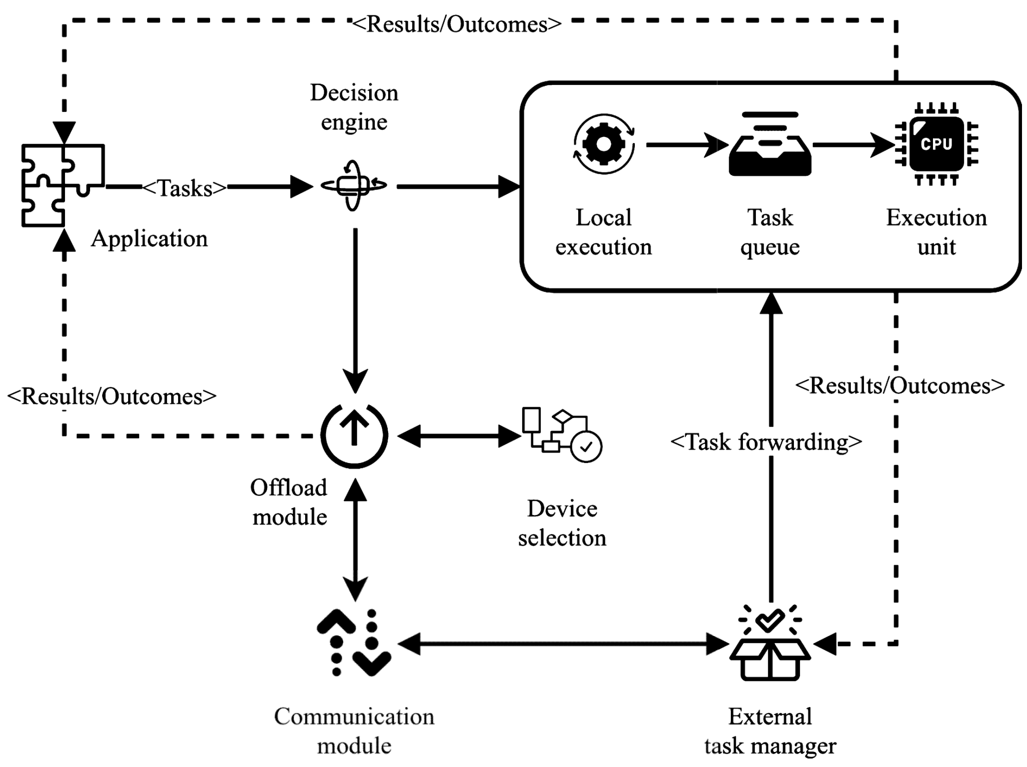


**Figure 3.1:** Architecture for the Vehicular Computational model

## 3.2 System Model

we consider a bi-directional road with two way traffic streams.A set of vehicles which are generating the Tasks is defined as$N$=1, 2, 3....$n$.Every Vehicle runs with uniform speed along the both two way road.Each vehicle is entrusted with a substantial computational workload, which can be carried out either locally by the vehicle's terminal.

### 3.2.1 Energy Model

Our primary focus lies in minimizing energy consumption, encompassing both the energy expended in local computations and the energy required for task offloading to neighboring vehicles.

### 3.2.2 Local Computation Energy

The energy consumed during local computations of Electric Vehicle $n$ in time slot $k$ can be quantified by the local computing time, $t^k_{n,l}$.where K is a Beginning time slot.

$$t_{n,l}^{k} = D_{n,l}^{k}\Omega/f - n$$

where $D_{n,l}^{k}$ is the size of Queue for local offloading. $f_n$ is the frequency of CPU and $\Omega$ shows number of cpu cycles in MIPS required to process one bit. The energy consumed by the Electric Vehicle $n$ during local computations in time slot $k$ can be derived by calculating the local computation energy, $E_{n,l}^{k}$.

$$E_{n,l}^{k} = \delta f^{3} - nt_{n,l}^{k}$$

where $\delta$ shows the computation energy efficiency coefficient.//

### 3.2.3 Vehicle to Vehicle Offloading Energy

The time taken for Electric vehicle $n$ to offload its tasks to a neighboring vehicle, denoted as $m$, in time slot $k$can be determined by the offloading time, $t_{n,m}^{k}$.

$$t_{n,m}^{k} = D_{n,m}^{k} \ / \ V_{n,m}^{k}$$

where,$D_{n,m}^{k}$ is the size of the queue of neighboring vehicle.$V_{n,m}^{k}$ represents the transmission rate between the sender and receiver electric vehicles.

The energy consumed by Electric Vehicle $n$ when offloading tasks to another neighboring Vehicle supposed $m$ in time slot $k$ can be expressed as the offloading energy, $E_{n,m}^{k}$.

$$E_{n,m}^{k} = F_{n}^{k}t_{n,m}^{k}$$

here $F$ is the transmission power of the Electric Vehicle. The total Energy of Electric Vehicle $n$ in $k$ time slot is the total Energy of the Local computation energy and vehicle-to-vehicle offloading energy.

$$E_{k}^{n} = E_{n,l}^{k} + E_{n,m}^{k}$$

### 3.2.4 Stream Model

The vehicle-to-vehicle (V2V) task offloading system for stream processing involves a network of vehicles $N$=$N1 \cup N_2 \cup N_3 ... N_n$ equipped with computational resources and communication capabilities. These vehicles move within a geographic area $A$=$A_1 \cup A_2 .. A_n$ and collect data from various sources, generating continuous data streams.

Each Electric vehicle in the system acts as a processing node, capable of performing computational tasks. The vehicles communicate with each other using V2V communication to exchange data and collaborate on stream processing tasks.

The system employs a **Distributed Stream Processing** approach, where each vehicle $n$ processes a portion of the task it collects, utilizing its available computational resources. However, when a vehicle encounters computationally intensive tasks or lacks sufficient resources to process the tasks, it can offload those tasks to neighboring vehicles $m$ with more resources or higher processing capabilities.

Once a suitable vehicle for offloading is identified, the task is transferred from the offloading vehicle $n$ to the receiving vehicle$m$ over the V2V communication link. The receiving vehicle then performs the offloaded task using its available resources and returns the results to the requesting vehicle. Overall, our system model for V2V task offloading through stream processing enables vehicles to collaborate and optimize resource usage, enhancing the scalability, responsiveness, and energy efficiency of stream processing tasks in dynamic vehicular environments.

**Algorithm 1** Stream Processing Algorithm

```
$V$: Collection of Electric Vehicles
$Task$: Task generated by a Vehicle
$distance$: Range of vehicles for offloading tasks
$V.Px$: Current value of the X direction
$V.Py$: Current value of the Y direction
$X$: X-axis Direction of the Vehicle
$Y$: Y-axis Direction of the Vehicle
$angle$: Calculated angle
procedure ClusterFormation
    while true do
        if $d < 50$ then
            Send the $Task$ to vehicle $V$
        end if
        Initialize $X$ and $Y$ by getting the X-axis and Y-axis values of the vehicles
        if $X > V.Px$ then
            Calculate the angle between $X - V.Px$ and $V.Py - Y$ $X < V.Px$
            Set the angle in the range between 180 to 360 degrees.
        else
            Set the angle to 0
        end if
        if $45 \leq$ angle $< 135$ then
            Set the direction as LEFT $135 \leq$ angle $< 225$
            Set the direction as RIGHT $225 \leq$ angle $< 315$
            Set the direction as UP
        else
            Set the direction as DOWN
        end if
        Update the values of $X$ and $Y$
    end while
end procedure
```

Chapter 4

# Implementation and Results

We will delve into the minute details of the implementation process and the results obtained in this section of the chapter. The Anylogic simulator, a versatile piece of software that is ideal for modeling and simulating complex systems, was used to build and run the simulation itself. To ensure optimal performance and accuracy during the simulation, the system used for this research was outfitted with specific hardware requirements. The system's impressive 8GB of Random Access Memory (RAM) made it possible to handle large amounts of data and computations with ease.

The system also included a large 1TB Hard Disk Drive (HDD), which offered plenty of storage space for the simulation data, input parameters, and interim results, enabling easy access to information throughout the experiment. A powerful 2.6GHz Intel Core i7 processor was at the system's core to further increase processing power. This powerful processor made it possible to run complex mathematical simulations and algorithms quickly and accurately, which improved the research's overall effectiveness and dependability.

It should be noted that Windows was used as the operating system on this system. Windows is a popular and reliable platform known for its user-friendly interface and compatibility with a wide range of software applications. The accuracy, stability, and smooth operation of the simulation throughout the research project were all made possible by the choice of such an optimal hardware and software configuration.

Java was chosen as the programming language to use for this research project. Due to its adaptability, portability, and robustness, Java, a very well-liked and frequently used programming language, was specifically chosen as the best option for performing intricate and sophisticated tasks. Because Java is object-oriented, the researchers were able to design and organize the code in a way that would make it simple to maintain, reuse code, and scale. The robust typing system and explicit memory management of the language also ensured that potential errors were discovered early in the development cycle, improving the codebase's overall dependability and stability.

Java's platform independence is among its many important benefits. The Java Virtual Machine, which served as a bridge between the code and the underlying operating system, allowed the Java code created for this research to run without modification on a variety of platforms. It was simpler to share and replicate the research findings across a range of hardware and operating systems thanks to this cross-platform compatibility. The researchers also had access to a wide range of tools and resources to speed up development and simplify the implementation of complex functionalities thanks to Java's extensive standard library and the availability of numerous third-party libraries and frameworks. Java provided a rich set of libraries to meet various research requirements, whether it be for data processing, networking, or graphical user interface development.The language was a wise choice for handling sensitive data and reducing potential security vulnerabilities due to its emphasis on security through features like automatic garbage collection and sandboxing.

A thorough evaluation of the traditional approach's performance is shown in Figure 4.1, which shows the number of tasks created and carried out in relation to various vehicle densities. The trends seen during the experimentation process are highlighted in the figure using a simple and instructive representation. The figure's x-axis represents vehicle density, and the y-axis represents the number of tasks. The rate of task execution noticeably decreases as the number of tasks rises, demonstrating a connection between task generation and execution effectiveness. Several insights can be drawn from the data points after careful examination. A total of 703 tasks were generated at a vehicle rate of 50 tasks, of which 697 were completed successfully. The number of tasks generated increased to 1145 at a vehicle rate of 100 tasks, with 1135 of those tasks being executed proficiently. The total number of generated tasks increased to 1871 as the vehicle rate was increased to 150 tasks per minute. Of these, 1852 tasks were completed on time. The number of tasks generated later increased to 2382 at a vehicle rate of 200 tasks, and impressively, 2357 of those tasks were carried out without much delay. Last but not least, the highest number of generated tasks—3067 in total—was recorded when the vehicle rate reached 250 tasks. Notably, a sizable portion of these tasks—more specifically, task 3034—were successfully completed. In the context of the conventional approach, the data presented offers useful insights into the relationship between vehicle density, task generation, and task execution rates. These results provide essential data for evaluating the system's performance under various circumstances and can be used as a foundation for further optimization and development of the strategy.

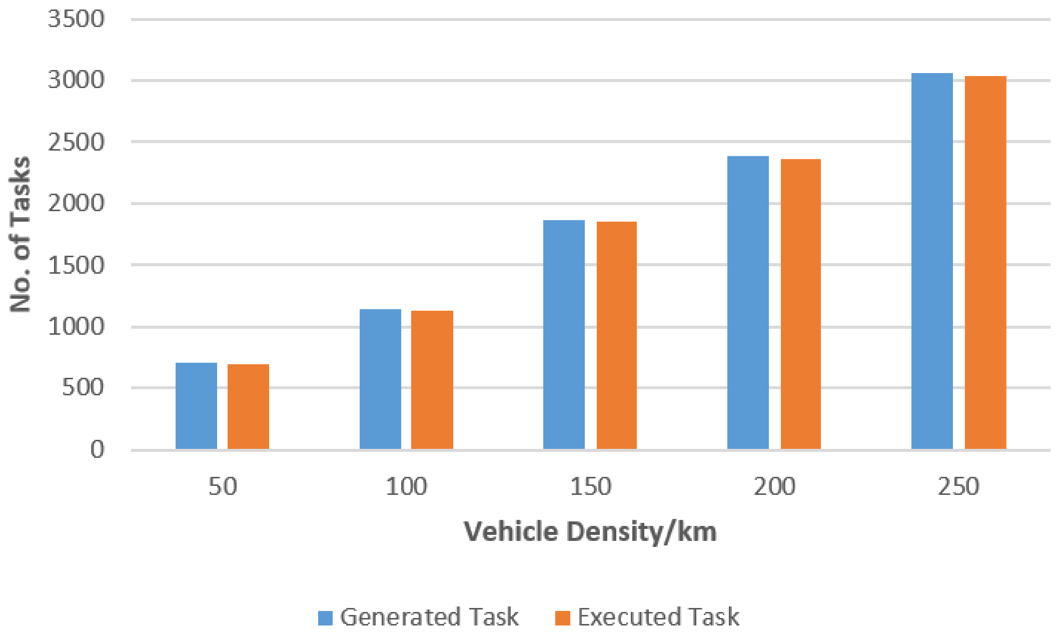


**Figure 4.1:** Tasks Generated vs Tasks Executed

Figure 4.2 meticulously illustrates the performance evaluation of the novel approach used in this study and provides a detailed comparison of tasks that were executed and tasks that were created using this cutting-edge approach. The performance of the new approach is noticeably better than that of the conventional approach, demonstrating its potential as a more effective and efficient solution for the task at hand. The vehicle density is shown on the x-axis in Figure 4.2 while the number of tasks is shown on the y-axis. According to the data, when the number of tasks rises, the task execution rate falls, mirroring the behavior seen with the conventional approach. By analyzing the specific data points, we can see that a total of 611 tasks were generated at a vehicle rate of 50 tasks, and an impressive 610 of those tasks were actually completed, demonstrating a high level of success and proficiency. The number of tasks generated increased to 1318 when the vehicle rate was raised to 100 tasks, and 1315 of those tasks were completed with impressive efficiency. When the assessment was run at a vehicle rate of 150 tasks per minute, the new methodology produced 1958 tasks, of which 1948 were completed, supporting the method's improved performance. With a vehicle rate of 200 tasks, the new approach generated 2577 tasks, and impressively, 2760 of those tasks were completed successfully, demonstrating its ability to handle larger task volumes with faster execution rates. Last but not least, the highest number of tasks—3208 overall—was generated at a vehicle rate of 250 tasks. The new approach's ability to consistently deliver exceptional performance was further demonstrated by the impressive 3190 tasks that were successfully completed. The data in Figure reff-vd2 is strong support for the claim that the new approach performs better than the conventional method at completing tasks. This insightful information paves the way for future advancements and field-specific improvements as the new method shows promise in delivering improved performance and efficiency across a range of vehicle densities and task scenarios.

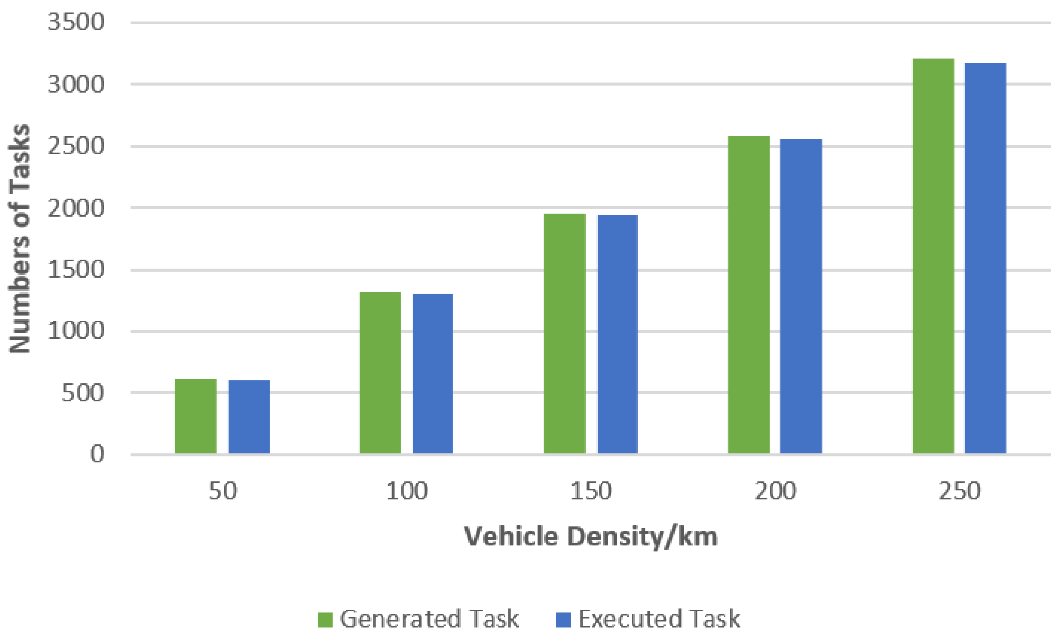


**Figure 4.2:** Tasks Generated vs Tasks Executed using New Approach

Figure 4.3 presents a thorough analysis of the task delivery rate and the corresponding failure rate, offering insightful information about how the system performs when there are different densities of vehicles. The vehicle density is shown on the x-axis, and the number of tasks is shown on the y-axis. Intriguing correlations between the failure rate and the vehicle arrival rate, as well as between the task delivery rate and the vehicle arrival rate, are shown in the graph. The failure rate exhibits a discernible upward trend as the vehicle arrival rate rises, indicating a potential challenge in effectively managing tasks when the system experiences increased traffic. In contrast, a declining pattern in the task delivery rate is seen as the vehicle delivery rate rises. This suggests that the ability to complete tasks on time decreases as the system is overloaded with more vehicle arrivals. These observations are confirmed by a closer look at the data points in the figure: The system received a total of 703 tasks at a vehicle arrival rate of 50. Of these, 635 tasks were successfully completed, while 68 tasks—representing a negligible portion of the total tasks—were unsuccessful. The number of tasks received increased to 1145 as the vehicle arrival rate reached 100. Out of these, 924 tasks were successfully completed, but 221 tasks failed, indicating that the failure rate increases as the system is subjected to more traffic. A greater influx of tasks was observed at a vehicle arrival rate of 150, with 1872 tasks being received. However, there were difficulties in completing these tasks, leading to 1311 successful completions and 561 marked as failures. Continuing the evaluation, the system received 2382 tasks at a vehicle arrival rate of 200. While 1015 tasks were flagged as failures, 1367 tasks were successfully delivered despite a decline in delivery performance. The highest number of received tasks, 3067 overall, was noticed when the vehicle arrival rate reached 250. Although 1396 tasks were successfully delivered despite a significant number of 1671 tasks failing, it is regrettable that the task delivery rate was under a lot of stress. The data shown offers insightful information about the system's performance in terms of task completion and failure rates at various vehicle densities. This thorough analysis emphasizes the demand for additional research and possible optimization techniques to boost task delivery effectiveness and reduce failure rates, particularly in scenarios with higher vehicle arrival rates.

Figure 4.4 provides a thorough analysis of the task completion rate and corresponding failure rate using our novel methodology. The x-axis in the figure represents vehicle density, while the y-axis shows the number of tasks, clearly demonstrating how the system performs under various vehicle densities. The notable decrease in the failure rate shows how significantly better our approach is than the conventional approach. In fact, it appears that the failure rate has been reduced to the point where no task fails, which represents a significant improvement in the effectiveness and dependability of the system. The analysis of the particular data points in the figure confirms these

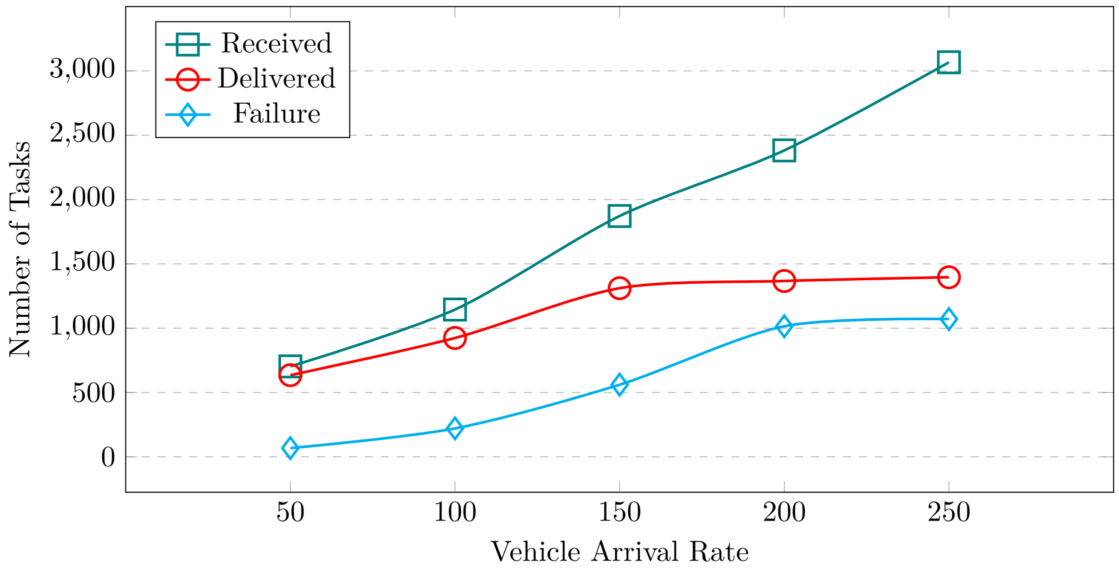


**Figure 4.3:** Tasks Delivery and Failure Count using Traditional Approach

encouraging findings: A total of 628 tasks were received by the system at a vehicle arrival rate of 50, demonstrating the robustness and efficacy of our strategy. Impressively, all 628 tasks were completed successfully, with no instances of failure. As the assessment went on, 1331 tasks were received when the vehicle arrival rate reached 100. All but one task was completed successfully, mirroring the exceptional performance, resulting in a very low failure rate that is practically zero. The system was presented with 1958 tasks at a vehicle arrival rate of 150. Our method once again proved its worth by successfully completing 1888 tasks; this further validates the method's capacity to handle increased task volumes with unmatched success. The system then received 2577 tasks when the vehicle arrival rate reached 200, and the majority of these tasks (2478) were also completed successfully and without any errors, supporting the consistent and dependable performance of our approach. The system received 3208 tasks at the highest vehicle arrival rate of 250, and all 3067 of them were successfully completed without any errors. This outstanding accomplishment demonstrates the method's adaptability and resilience even in difficult circumstances. The data presented unequivocally demonstrates our approach's superiority to the conventional approach in terms of task delivery and failure rates. Our method has the potential to handle tasks with exceptional reliability and success, even in situations with high vehicle arrival rates, as evidenced by the significant decrease in failure rates, which are practically approaching zero. These striking outcomes open the door for future developments and practical applications of our strategy.

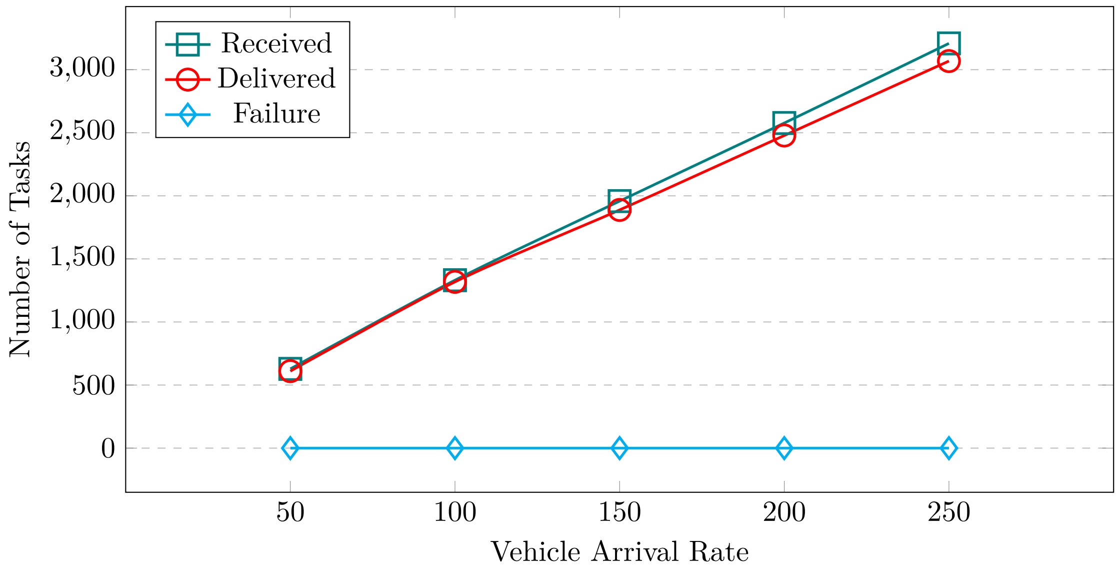


**Figure 4.4:** Tasks Delivery and Failure Count using Our Approach

The average waiting times for tasks under various vehicle rates are shown in detail in Figure 4.5 in milliseconds. The graph explains the connection between the vehicle rate and the amount of time tasks wait in the system before being carried out. The observed average waiting time for a task was 163 milliseconds at a vehicle rate of 50. This suggests that tasks were processed and executed quickly on average after being added to the queue. The average wait time for tasks then increased to 349 milliseconds as the vehicle rate reached 100. This extended waiting time is a result of the system's reduced ability to quickly process and complete incoming tasks due to higher vehicle rates. The average task waiting time increased further to 522 milliseconds at a vehicle rate of 150. This demonstrates how the system is under increasing pressure as vehicle rates increase, leading to longer wait times for tasks to be completed. Continuing the analysis, the average task waiting time increased to 686 milliseconds when the vehicle rate reached 200. The system's capacity limitations under higher vehicle rates are indicated by the system's significant increase in waiting times. Last but not least, the task waiting time averaged 852 milliseconds at the highest vehicle rate of 250. This lengthy wait time highlights the system's difficulties in managing a heavy influx of tasks at faster vehicle rates. The information provided offers useful insights into the typical wait times for tasks at various vehicle rates. The average waiting time for tasks noticeably grows along with vehicle rate, indicating potential constraints on the system's ability to complete tasks quickly. These results provide essential knowledge for system performance optimization,

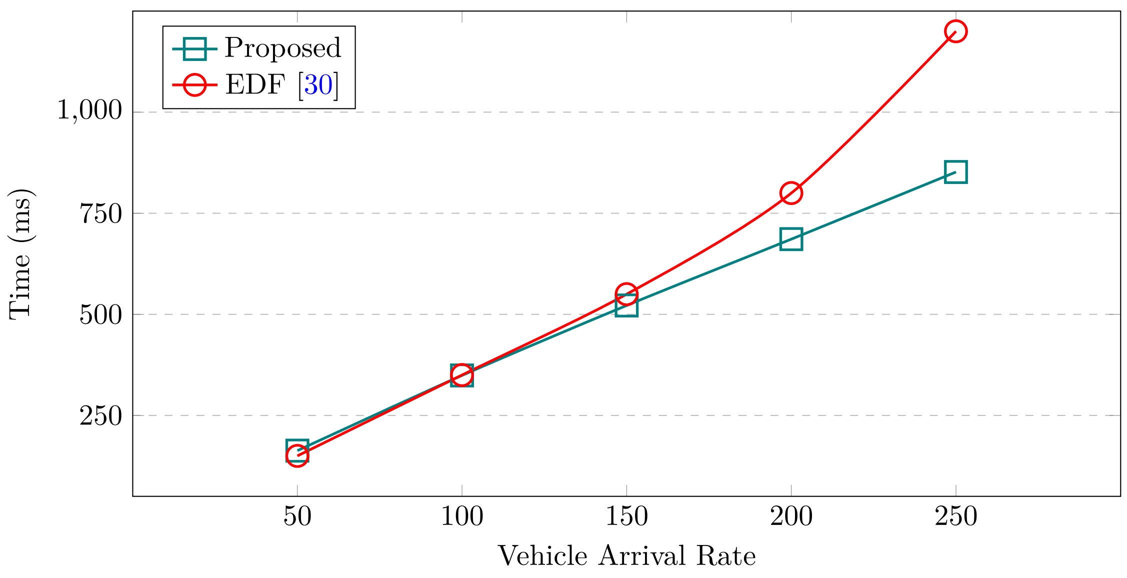


**Figure 4.5:** Average Waiting Time

ensuring a balance between vehicle rates and task processing capabilities to reduce wait times and increase overall efficiency.

In Figure 4.6, we are comparing the technique we have put forward with the method employed in the study [30], which involves the utilization of the Earliest Deadline First (EDF) technique. Our approach demonstrates enhanced accuracy and task delivery ratio in contrast to the EDF technique. Specifically, we observe a significant decrease in the task delivery ratio of the EDF technique as the vehicle arrival rate is increased, whereas our technique maintains a higher task delivery ratio under similar conditions.

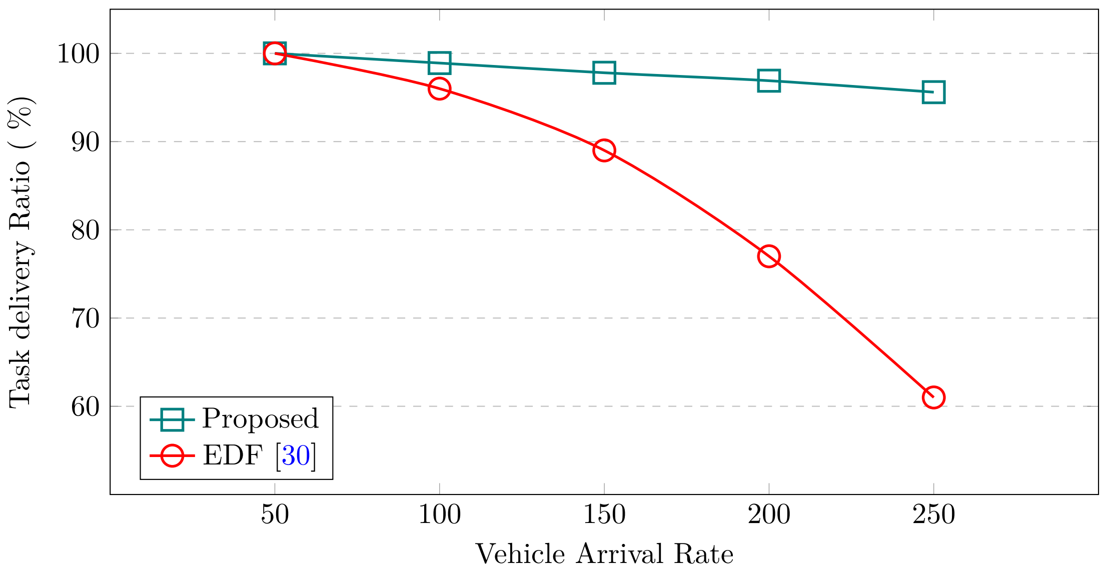


**Figure 4.6:** Task Delivery Ratio with the Increasing Number of Vehicles

Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

this thesis has explored a novel and innovative approach to address the challenges posed by real-time stream data processing in resource-constrained environments. The integration of EVs as computational resources within stream processing systems has been thoroughly investigated and demonstrated to offer significant enhancements in scalability, efficiency, and overall system performance. Through a comprehensive review of existing literature, we have established the context and importance of real-time stream data processing, highlighting the limitations of traditional approaches in accommodating the continuous influx of data streams. The emergence of the IoT and the exponential growth of data have necessitated a paradigm shift, and our proposed framework has responded to this need by capitalizing on the computational capabilities and energy storage of EVs. The design and methodology section of this thesis presented a systematic and detailed framework for EV-assisted resource sharing in stream data processing systems. The approach leverages V2V communication to dynamically offload computational tasks and optimize resource utilization. A comprehensive network model and system model were formulated to facilitate the integration of EVs as processing nodes, considering factors such as energy consumption, task execution times, and offloading strategies. The implementation and results section provided an in-depth analysis of the proposed approach through extensive simulations and performance evaluations. The comparison between the novel approach and conventional methods showcased the significant advantages offered by our framework. The novel approach consistently demonstrated higher task completion rates, markedly reduced failure rates, and optimized energy consumption. These results underscore the effectiveness and potential of integrating EVs into stream processing systems, especially in dynamic vehicular environments. In essence, this thesis contributes to the fields of stream data processing, edge computing, and vehicular networks by presenting a pioneering solution to the challenges of real-time data analysis. By tapping into the computational and energy resources of EVs, our approach offers a scalable, efficient, and sustainable solution for processing continuous data streams. The findings of this thesis highlight the transformative potential of EVs as computational nodes in stream data processing systems, showcasing their ability to revolutionize the landscape of data processing and decision-making in the era of IoT.

## 5.2 Future Work

As we look to the future, further research can explore additional dimensions of this approach, such as optimizing task allocation algorithms, investigating the impact of varying EV densities, and considering the implications of different communication technologies. Additionally, the integration of machine learning and artificial intelligence techniques can enhance the adaptability and intelligence of the proposed framework. Ultimately, this thesis sets the stage for continued exploration and innovation in the field, with the potential to reshape the way real-time data streams are processed and harnessed for actionable insights.

# Bibliography

[1] Seref Sagiroglu and Duygu Sinanc. “Big data: A review”. In: *2013 international conference on collaboration technologies and systems (CTS)*. IEEE. 2013, pp. 42–47.

[2] Diego Garcıa-Gil et al. “A comparison on scalability for batch big data processing on Apache Spark and Apache Flink”. In: *Big Data Analytics* 2.1 (2017), pp. 1–11.

[3] Matteo Muratori et al. “The rise of electric vehicles—2020 status and future expectations”. In: *Progress in Energy* 3.2 (2021), p. 022002.

[4] Saeed Shahrivari. “Beyond batch processing: towards real-time and streaming big data”. In: *Computers* 3.4 (2014), pp. 117–129.

[5] Kamalika Dutta and Manasi Jayapal. “Big data analytics for real time systems”. In: *Big Data analytics seminar*. 2015, pp. 1–13.

[6] Jane Lubchenco and Peter M Haugan. “Technology, Data and New Models for Sustainably Managing Ocean Resources”. In: *The Blue Compendium: From Knowledge to Action for a Sustainable Ocean Economy*. Springer, 2023, pp. 185–211.

[7] Matthew T Lawder et al. “Battery energy storage system (BESS) and battery management system (BMS) for grid-scale applications”. In: *Proceedings of the IEEE* 102.6 (2014), pp. 1014–1030.

[8] Rajkumar Buyya et al. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”. In: *Future Generation computer systems* 25.6 (2009), pp. 599–616.

[9] Oladayo Bello and Sherali Zeadally. “Intelligent device-to-device communication in the internet of things”. In: *IEEE Systems Journal* 10.3 (2014), pp. 1172–1182.

[10] Donghyeog Choi, Hyunchul Choi, and Donghwa Shon. “Future changes to smart home based on AAL healthcare service”. In: *Journal of Asian Architecture and Building Engineering* 18.3 (2019), pp. 190–199.

[11] Talal Ashraf Butt et al. “Social Internet of Vehicles: Architecture and enabling technologies”. In: *Computers & Electrical Engineering* 69 (2018), pp. 68–84.

[12] Elarbi Badidi and Karima Moumane. “Enhancing the processing of healthcare data streams using fog computing”. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2019, pp. 1113–1118.

[13] Neha, Pooja Gupta, and MA Alam. “Challenges in the adaptation of IoT technology”. In: *A Fusion of Artificial Intelligence and Internet of Things for Emerging Cyber Systems* (2022), pp. 347–369.

[14] Abdelkarim Ben Sada et al. “A distributed video analytics architecture based on edge-computing and federated learning”. In: *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE. 2019, pp. 215–220.

[15] Kashif Bilal et al. “Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers”. In: *Computer Networks* 130 (2018), pp. 94–120.

[16] Yuan Ai, Mugen Peng, and Kecheng Zhang. “Edge computing technologies for Internet of Things: a primer”. In: *Digital Communications and Networks* 4.2 (2018), pp. 77–86.

[17] Hesham El-Sayed et al. “Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment”. In: *ieee access* 6 (2017), pp. 1706–1717.

[18] Ricardo S Alonso et al. “An intelligent Edge-IoT platform for monitoring livestock and crops in a dairy farming scenario”. In: *Ad Hoc Networks* 98 (2020), p. 102047.

[19] Ejaz Ahmed and Mubashir Husain Rehmani. *Mobile edge computing: opportunities, solutions, and challenges.* 2017.

[20] Sonia Shahzadi et al. “Multi-access edge computing: open issues, challenges and future perspectives”. In: *Journal of Cloud Computing* 6 (2017), pp. 1–13.

[21] Huan Zhou et al. “A survey on mobile data offloading technologies”. In: *IEEE access* 6 (2018), pp. 5101–5111.

[22] Filippo Rebecchi et al. “Data offloading techniques in cellular networks: A survey”. In: *IEEE Communications Surveys & Tutorials* 17.2 (2014), pp. 580–603.

[23] Junaid Qadir et al. “Towards mobile edge computing: Taxonomy, challenges, applications and future realms”. In: *Ieee Access* 8 (2020), pp. 189129–189162.

[24] Abhishek Kumar Gaurav et al. “A survey on computation resource allocation in IoT enabled vehicular edge computing”. In: *Complex & Intelligent Systems* 8.5 (2022), pp. 3683–3705.

[25] Kate Pangbourne et al. “Questioning mobility as a service: Unanticipated implications for society and governance”. In: *Transportation research part A: policy and practice* 131 (2020), pp. 35–49.

[26] Junhui Zhao et al. “Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks”. In: *IEEE Transactions on Vehicular Technology* 68.8 (2019), pp. 7944–7956.

[27] Paris Carbone et al. “Apache flink: Stream and batch processing in a single engine”. In: *The Bulletin of the Technical Committee on Data Engineering* 38.4 (2015).

[28] Muhammad Hussain Iqbal, Tariq Rahim Soomro, et al. “Big data analysis: Apache storm perspective”. In: *International journal of computer trends and technology* 19.1 (2015), pp. 9–14.

[29] Giuseppe Aceto, Valerio Persico, and Antonio Pescapé. “Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0”. In: *Journal of Industrial Information Integration* 18 (2020), p. 100129.

[30] Tariq Qayyum et al. “Multi-level resource sharing framework using collaborative fog environment for smart cities”. In: *IEEE access* 9 (2021), pp. 21859–21869.

Appendix A

# First Appendix

The separate numbering of appendices is also supported by LaTeX. The *appendix* macro can be used to indicate that following chapters are to be numbered as appendices. Only use the *appendix* macro once for all appendices.